WebAssembly動かしてみた

Keiya Nobuta

自己紹介

- 信田圭哉
- 組込み向けLinuxの開発/メンテナンス
- AGLなどでも活動してます

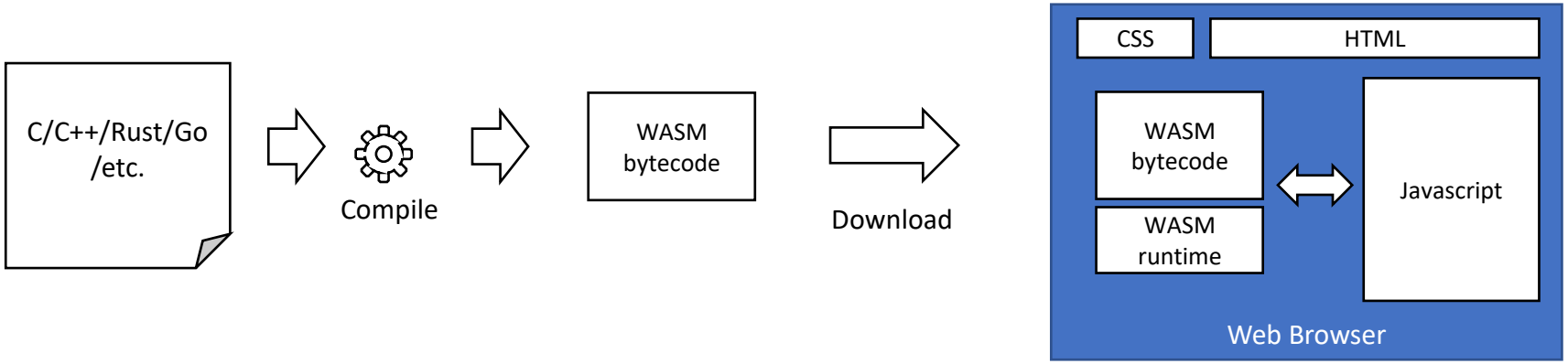
WebAssembly (WASM) って?

- スタック ベースで動作する仮想マシン用のバイト・コードの仕様
  - Javascriptと連携して動作し、ネイティブ コードと同等の速度を出すことを目指して作られた
- 仕様がすべてオープン
- 主要なWebブラウザーが対応
  (Firefox, Chrome, Edge, Safari, Opera)

WASMのざっくりとした歴史

- 2015年、仕様の策定を行うW3Cのコミュニティ グループの一つとしてWebAssembly Community Groupが発足
- 2017年ごろWebブラウザーでの実装が公開

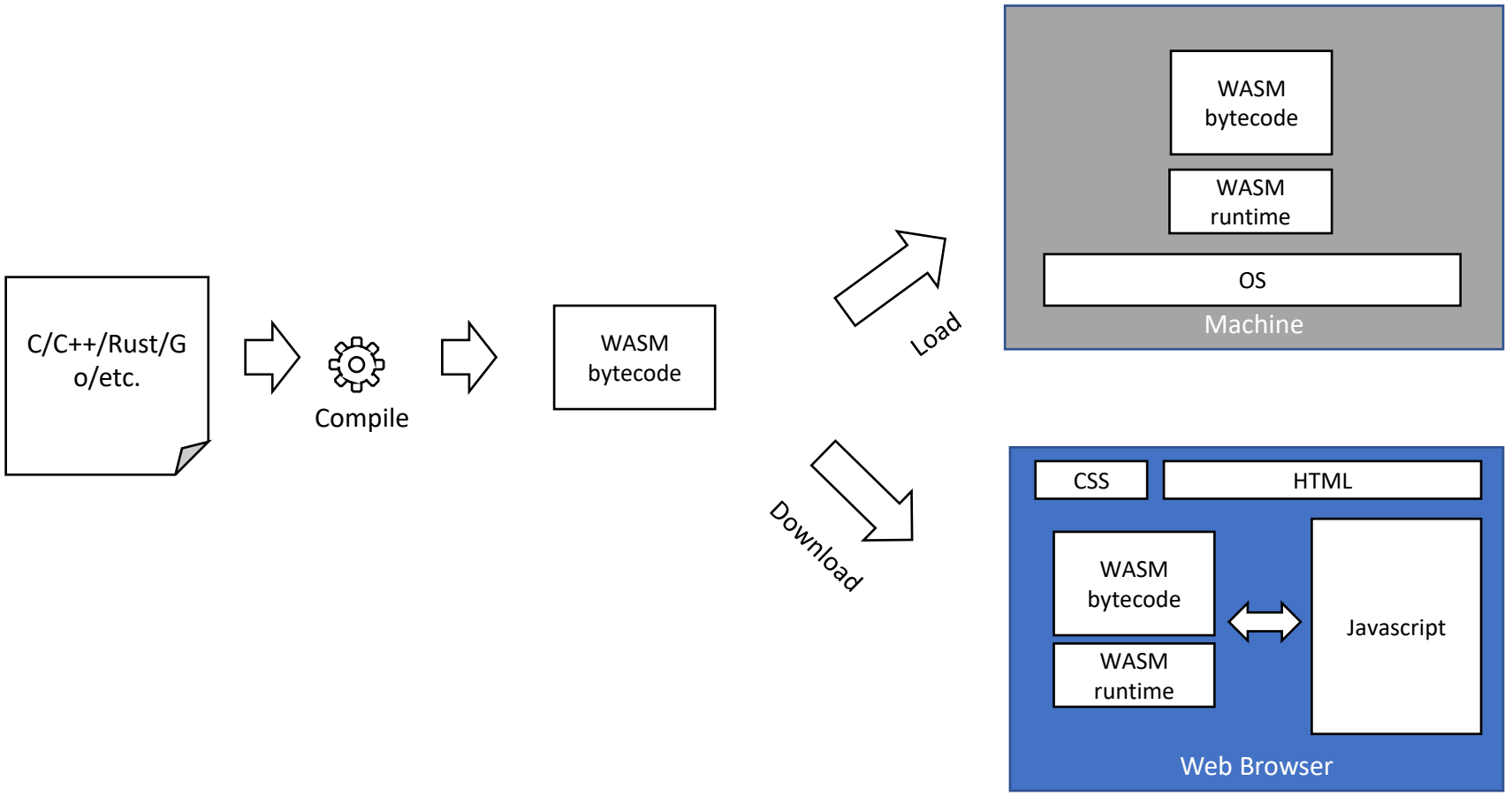
イメージ


C/C++/Rust/Go
/etc.
Compile
WASM
bytecode
Download
CSS
HTML
WASM
bytecode
WASM
runtime
Javascript
Web Browser

WebAssembly System Interface (WASI)

- Webブラウザー以外(サーバー サイド)でWASMを動作させるための仕様
- OSの機能を利用するためのインターフェースを定義することで、特定のOSに依らずすべての異なるOSで実行できる
- 仕様はすべてオープン
- VM (WASM Runtime) の実装がすでにいくつかある

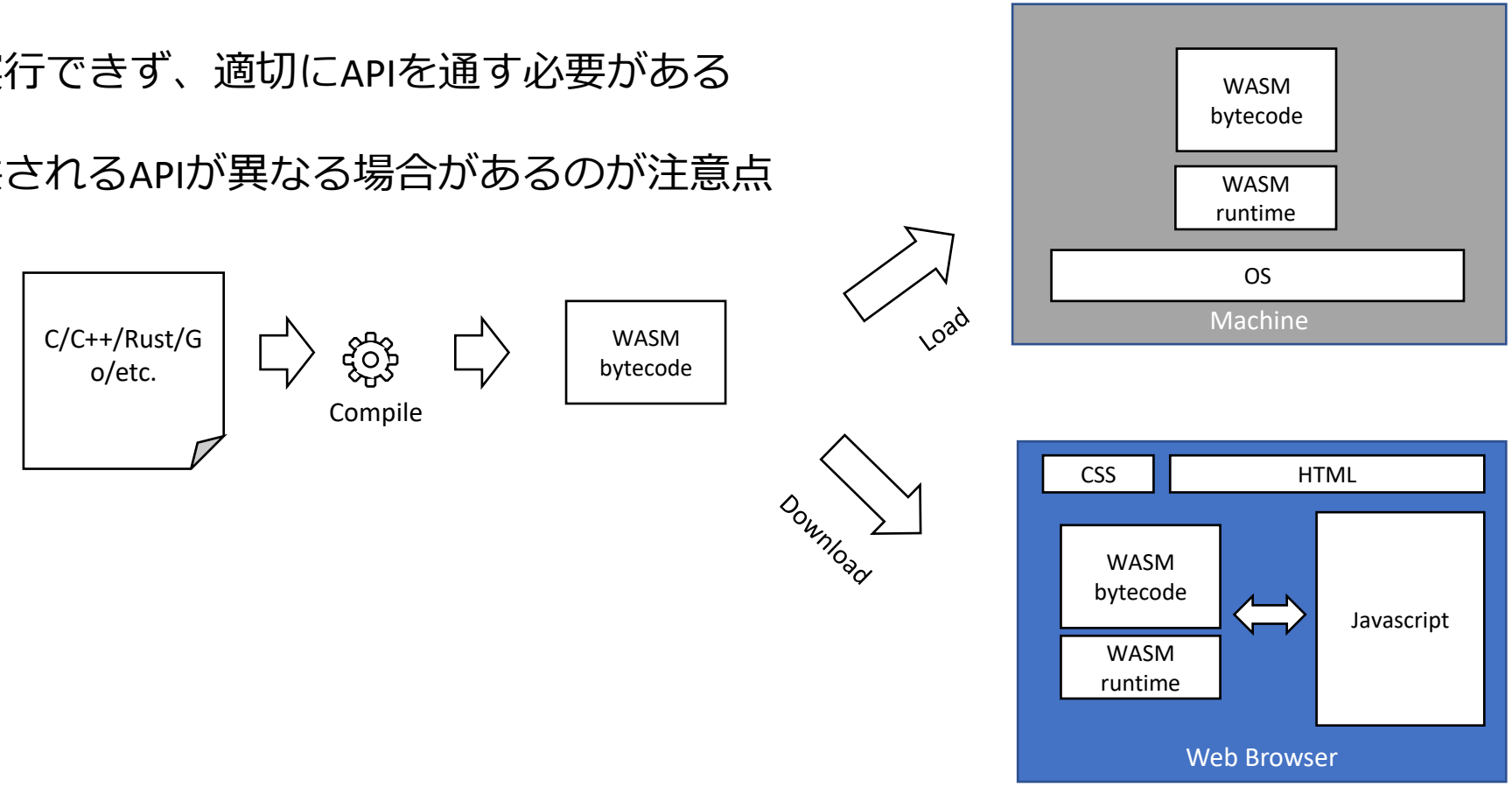
イメージ


C/C++/Rust/Go/etc.
Compile
WASM bytecode
Load
Download
WASM bytecode
WASM runtime
OS
Machine
CSS
HTML
WASM bytecode
WASM runtime
Javascript
Web Browser

WASMの特徴

- C/C++をはじめ、Rust、Goなど様々な言語からコンパイル可能
- ポータブル
  - ランタイムの仕様がオープン、様々なプラットフォームに対応可能
- サンドボックス化された実行環境
  - 高いメモリ安全性
  - ホストのコードは直接実行できず、適切にAPIを通す必要がある

※ ランタイムの実装ごとに提供されるAPIが異なる場合があるのが注意点



WASIのランタイム実装はいくつかある

- Wasmtime
  - リファレンス実装的な位置づけ
- WebAssembly Micro Runtime (WAMR)
  - 軽量、ポータビリティ重視
- Wasmer
  - コンパイラ、ランタイム、性能測定や制御など様々な機能を提供
  - wapmというWASMパッケージ マネージャーとの連携が強い

今回使ってみたのはコレ

WAMRの特徴

- ソース:
  - GitHub - bytecodealliance/wasm-micro-runtime: WebAssembly Micro Runtime (WAMR)
- ランタイムのバイナリが軽量、メモリ使用量も抑えている
- 様々なプラットフォームに対応
  - Linux、Windows等の汎用OS
  - Android, iOSなどのスマートフォン向け
  - NuttX, Zephyr, VxWorksなどのRTOS

動かしてみる (1)

- Linux向けにランタイムをビルド
  - build直下に'iwasm'が生成される
    (とりあえずテストするだけなのでここにPATHを通しておく)

```
git clone https://github.com/bytecodealliance/wasm-micro-runtime.git
cd wasm-micro-runtime/product-mini/platforms/linux
mkdir build && cd build
cmake ..
make
<...>
export PATH=$PWD:$PATH
```

- サンプルのWASMバイナリをビルド
  - ※ドキュメントを参考に、WASI SDKなどのビルド ツールをあらかじめインストールしておく

```
cd ../../../app-samples/hello-world
./build.sh
<...>
```

```
/*
 * Copyright (C) 2019 Intel Corporation.  All rights reserved.
 * SPDX-License-Identifier: Apache-2.0 WITH LLVM-exception
 */

#include <stdio.h>
#include <stdlib.h>

int
main(int argc, char **argv)
{
    char *buf;

    printf("Hello world!¥n");

    buf = malloc(1024);
    if (!buf) {
        printf("malloc buf failed¥n");
        return -1;
    }

    printf("buf ptr: %p¥n", buf);

    snprintf(buf, 1024, "%s", "1234¥n");
    printf("buf: %s", buf);

    free(buf);
    return 0;
}
```

- 実行

```
iwasm test.wasm
Hello world!
buf ptr: 0x1460
buf: 1234
```

動かしてみる (2)

- RustでHello World

```
fn main() {
    println!("Hello world!");
}
```

```
cargo build --target wasm32-wasi
```

```
iwasm target/wasm32-wasi/debug/helloworld.wasm
```

```
Hello world!
```

動かしてみる (3)

- Zephyr
  - riscv32のqemu向けにビルド

```
export ZEPHYR_BASE=$HOME/zephyrproject/zephyr
cd product-mini/platforms/zephyr/simple
cp boards/qemu_riscv32.conf prj.conf
west update
<...>
./build_and_run.sh qemu_riscv32
```

```
-- west build: making build dir /data/github/bytecodealliance/wasm-micro-runtime/product-mini/platforms/zephyr/simple/build pristine
-- west build: generating a build system
<...>
[156/156] Linking C executable zephyr/zephyr.elf
Memory region         Used Size  Region Size  %age Used
           RAM:      114194 B       256 MB      0.04%
      IDT_LIST:         0 GB         2 KB      0.00%
-- west build: running target run
[0/1] To exit from QEMU enter: 'CTRL+a, x'[QEMU] CPU: riscv32
*** Booting Zephyr OS build zephyr-v3.3.0-3099-gd2917610a5ba ***
Hello world!
buf ptr: 0xd8
buf: 1234
elpase: 30
```

Zephyr向け実装を見てみると…

product-mini/platforms/zephyr/simple/src/main.c

```
207       wasm_file_buf = (uint8 *)wasm_test_file;
208       wasm_file_size = sizeof(wasm_test_file);
209
210       /* load WASM module */
211       if (!(wasm_module = wasm_runtime_load(wasm_file_buf, wasm_file_size,
212                                             error_buf, sizeof(error_buf)))) {
213           printf("%s\n", error_buf);
214   #ifdef CONFIG_BOARD_ESP32
215           goto fail1 1;
```

product-mini/platforms/zephyr/simple/src/test_wasm.h

```
1   /*
2    * Copyright (C) 2019 Intel Corporation.  All rights reserved.
3    * SPDX-License-Identifier: Apache-2.0 WITH LLVM-exception
4    */
5
6   /**
7    * The byte array buffer is the file content of a test wasm binary file,
8    * which is compiled by wasi-sdk toolchain from C source file of:
9    *   product-mini/app-samples/hello-world/main.c.
10   */
11  unsigned char __aligned(4) wasm_test_file[] = {
12      0x00, 0x61, 0x73, 0x6D, 0x01, 0x00, 0x00, 0x00, 0x01, 0x10, 0x03, 0x60,
13      0x01, 0x7F, 0x01, 0x7F, 0x60, 0x02, 0x7F, 0x7F, 0x01, 0x7F, 0x60, 0x01,
14      0x7F, 0x00, 0x02, 0x31, 0x04, 0x03, 0x65, 0x6E, 0x76, 0x04, 0x70, 0x75,
15      0x74, 0x73, 0x00, 0x00, 0x03, 0x65, 0x6E, 0x76, 0x06, 0x6D, 0x61, 0x6C,
16      0x6C, 0x6F, 0x63, 0x00, 0x00, 0x03, 0x65, 0x6E, 0x76, 0x06, 0x70, 0x72,
17      0x69, 0x6E, 0x74, 0x66, 0x00, 0x01, 0x03, 0x65, 0x6E, 0x76, 0x04, 0x66,
18      0x72, 0x65, 0x65, 0x00, 0x02, 0x03, 0x02, 0x01, 0x01, 0x04, 0x05, 0x01,
```

サイズが…？

CのHello worldのWASMバイナリ	413B
RustのHello worldのWASMバイナリ	64KB

参考:

STM32 Nucleo L031K6	Flash ... 32KB
Raspberry Pi Pico	Flash ... 2MB SRAM ... 264KB

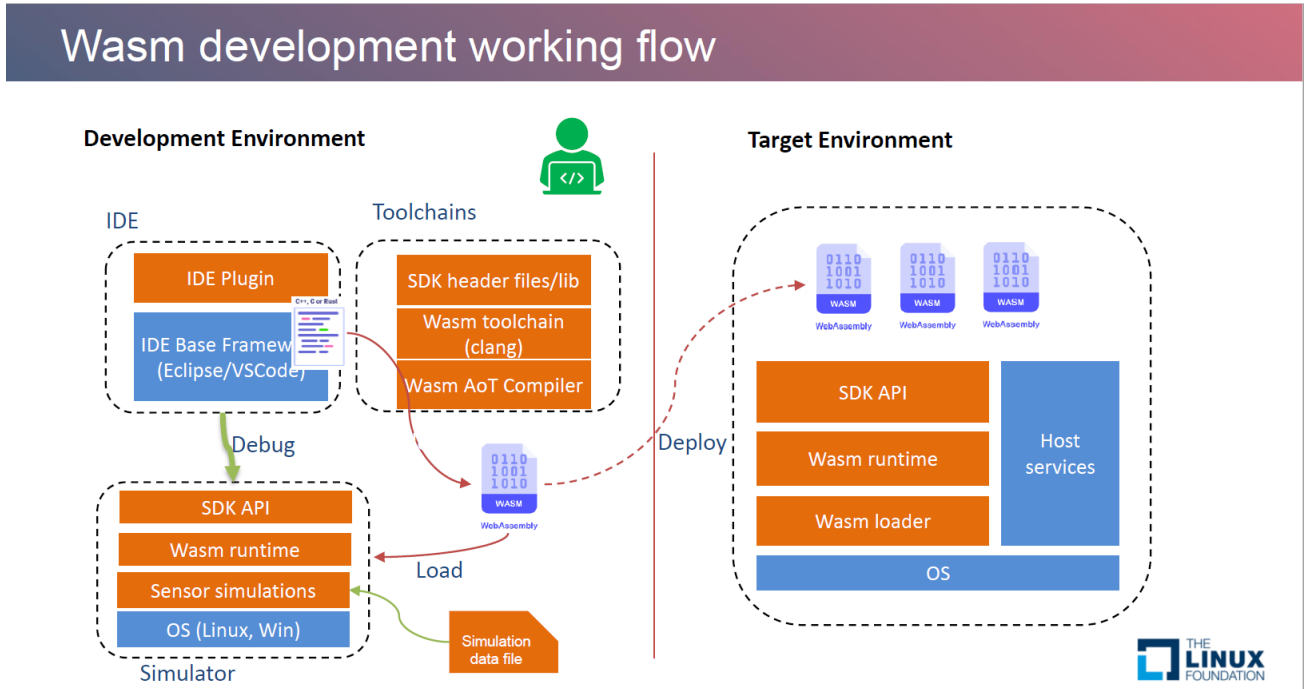
RTOS環境でWASM採用にアドバンテージはあるか？

- 特徴を改めて
  - 仕様・実装がオープンである
  - ポータブル、移植しやすい
  - セキュア
  - 様々な言語が利用可能

RTOS環境でWASM採用にアドバンテージはあるか？

- 特徴を改めて
  - 仕様・実装がオープンである
    ⇒ 特定のプラットフォーム ベンダーにロックインしない
  - ポータブル、移植しやすい
    ⇒ シミュレーション環境から実機へシームレスに移行可能、CI/CDにも有利
  - セキュア
    ⇒ OSコードをアプリから保護できる
    (手元では未検証)
  - 様々な言語が利用可能
    ⇒ アプリのエンジニア確保しやすさ

参考: Open Source Summit Japan & Automotive Linux Summit 2021: The Cool Features of WebAssembly Micro R... (sched.com)



現状の課題

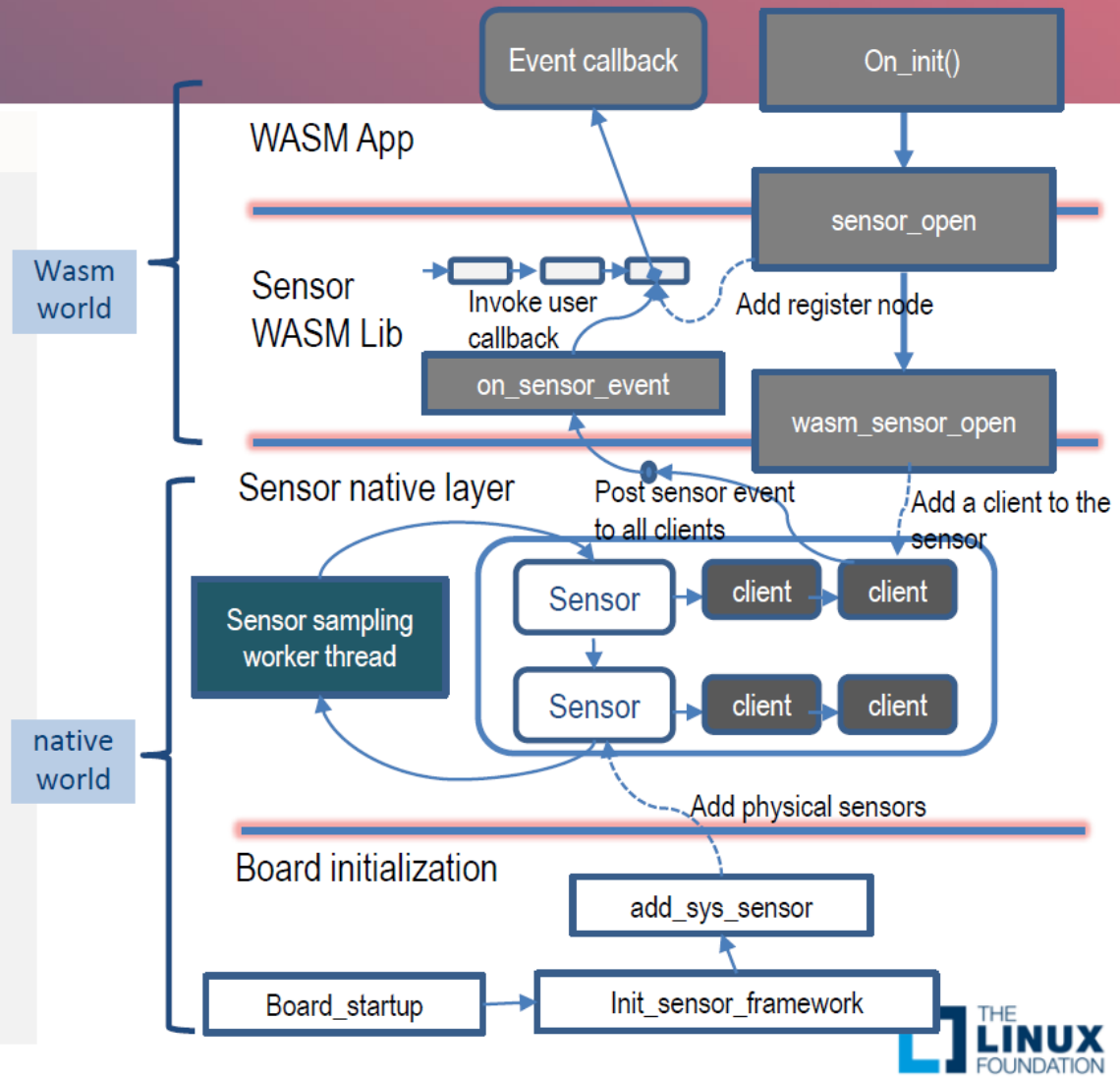
- 性能のチューニング幅は下がりそう
- 組込み特有の細かな制御の実現のためには、ある程度ランタイムの作りこみが必要
  - ポータビリティとのトレードオフ
  - アプリケーションのレイヤーでどこまで細かく操作させるべきか？
    - GPIO
    - I2C/SPI
    - MMIO

参考：WAMRのセンサー フレームワーク

Sensor API and sample

${wamr_root}/samples/simple/wasm-apps/sensor.c

```
#include "wasm_app.h"

/* Sensor event callback*/
Void sensor_event_handler(sensor_t sensor,
          attr_container_t *event,
          void *user_data) {
  printf("### app get sensor event\n");
  attr_container_dump(event);
}

Void on_init()
{
  sensor_t sensor;

/* open a sensor */
sensor = sensor_open("sensor_test", 0, sensor_event_handler, NULL);

  /* config the sensor */
  sensor_config(sensor, 2000, 0, 0);
}

void on_destroy() {
}
```



参考

- https://webassembly.org/
- https://www.w3.org/community/webassembly/
- Linuxコンテナの「次」としてのWebAssembly、の解説 (zenn.dev)
- WebAssemblyの歴史について (zenn.dev)
- Open Source Summit Japan & Automotive Linux Summit 2021: The Cool Features of WebAssembly Micro R... (sched.com)
- WASI (WebAssembly System Interface)のランタイム5種を動かす - Qiita

Thanks!